Freeform Search

Database:
- US Pre-Grant Publication Full-Text Database
- US Patents Full-Text Database
- US OCR Full-Text Database
- EPO Abstracts Database
- JPO Abstracts Database
- Derwent World Patents Index
- IBM Technical Disclosure Bulletins

Term: L1 and (control adj1 logic)

Display: 10 Documents in Display Format: KWIC Starting with Number 1

Generate: Hit List [✓ Hit Count] Side by Side Image

Search Clear Interrupt

Search History

DATE: Sunday, May 02, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
	DB=USPT; PLUR=YES; OP=ADJ		
L3	L1 and (instruction$ with (control adj1 logic))	1	L3
L2	L1 and (control adj1 logic)	1	L2
L1	6373848.pn.	1	L1

END OF SEARCH HISTORY

L12	L9 and (select$ with microengine$)	0	L12
L11	L9 and (select$ with (thread$))	4	L11
L10	L9 and (select$ with (mac$ or thread$))	58	L10
L9	L8 and (select$ with plurality with ports)	333	L9
L8	712/$.ccls. or 709/$.ccls.	23278	L8
L7	L6	1	L7
L6	L1 and ((select$ or assign$) with (process$ or instruction$ or subroutine$))	1	L6
L5	L1 and (process$ or instruction$ or subroutine$)	1	L5
L4	L1 and thread$	0	L4
L3	L1 and network	1	L3
L2	L1 and mac	1	L2
L1	6373848.pn.	1	L1

END OF SEARCH HISTORY

Refine Search

Search Results -

Term	Documents
THREADS	147811
THREAD	132668
(6 AND THREADS).USPT.	8
(L6 AND THREADS).USPT.	8

Database:
US Pre-Grant Publication Full-Text Database
[✓ US Patents Full-Text Database]
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L7

Refine Search

Recall Text Clear Interrupt

Search History

DATE: Sunday, May 02, 2004 Printable Copy Create Case

Set Name side by side	**Query**	**Hit Count**	**Set Name** result set
	DB=USPT; PLUR=YES; OP=ADJ		
L7	L6 and threads	8	L7
L6	L4 and (processor with programmable)	35	L6
L5	L4 and programmable	131	L5
L4	L1 and (media adj1 access adj1 controller)	244	L4
L3	L2 and MAC	8	L3
L2	L1 and (processor with (multiple or plurality) with programmable with engines)	10	L2
L1	709/$.ccls. or 712/$.ccls.	23278	L1

END OF SEARCH HISTORY

First Hit Fwd Refs

☐ Generate Collection

L9: Entry 2 of 5 File: USPT Nov 19, 2002

DOCUMENT-IDENTIFIER: US 6483804 B1
TITLE: Method and apparatus for dynamic packet batching with a high performance network interface

Detailed Description Text (5):
It should also be understood that the techniques of the present invention might be implemented using a variety of technologies. For example, the methods described herein may be implemented in software running on a programmable microprocessor, or implemented in hardware utilizing either a combination of microprocessors or other specially designed application specific integrated circuits, programmable logic devices, or various combinations thereof. In particular, the methods described herein may be implemented by a series of computer-executable instructions residing on a storage medium such as a carrier wave, disk drive, or other computer-readable medium.

Detailed Description Text (56):
As described above, the protocols corresponding to headers 212, 214 and 216 depend upon the network environment in which a packet is sent. The protocols also depend upon the communicating entities. For example, a packet received by a network interface may be a control packet exchanged between the medium access controllers for the source and destination computer systems. In this case, the packet would be likely to include minimal or no data, and may not include layer three protocol header 214 or layer four protocol header 216. Control packets are typically used for various purposes related to the management of individual connections.

Detailed Description Text (110):
In one embodiment of the present invention, header parser 106 parses a packet received from a network according to a dynamic sequence of instructions. The instructions may be stored in the header parser's instruction memory (e.g., RAM, SRAM, DRAM, flash) that is re-programmable or that can otherwise be updated with new or additional instructions. In one embodiment of the invention software operating on a host computer (e.g., a device driver) may download a set of parsing instructions for storage in the header parser memory.

Detailed Description Text (227):
Therefore, one manner of identifying the final portion of data in a flow's datagram is to examine the size of each packet and compare it to a figure (e.g., MTU) that a packet is expected to exceed except when carrying the last data portion. It was described above that control information is received by FDBM 108 from header parser 106. An indication of the size of the data carried by a packet may be included in this information. In particular, header parser 106 in one embodiment of the invention is configured to compare the size of each packet's data portion to a pre-selected value. In one embodiment of the invention this value is programmable. This value is set, in the illustrated embodiment of the invention, to the maximum amount of data a packet can carry without exceeding MTU. In one alternative embodiment, the value is set to an amount somewhat less than the maximum amount of data that can be carried.

Detailed Description Text (251):
In state 654, flow database manager 108 determines whether the data received with

the present packet appears to contain the final portion of data for the associated datagram/flow. As described in conjunction with state 620, this determination may be made on the basis of the amount of data included with the packet. If the amount of data is less than a threshold value (a programmable value in the illustrated embodiment), then no more data is expected and this is likely to be the only data for this flow. In this case the procedure continues at state 668. If, however, the data meets or exceeds the threshold value, in which case more data may be expected, the procedure proceeds to state 656.

Detailed Description Text (277):
With the resulting hash value, in state 708 a modulus operation is performed over the number of processors available for distributing or sharing the processing. Illustratively, software executing on the host computer (e.g., a device driver for NIC 100) programs or stores the number of processors such that it may be read or retrieved by load distributor 112 (e.g., in a register). The number of processors available for load balancing may be all or a subset of the number of processors installed on the host computer system. In the illustrated embodiment, the number of processors available in a host computer system is programmable, with a maximum value of sixty-four. The result of the modulus operation in this embodiment, therefore, is the number of the processor (e.g., from zero to sixty-three) to which the packet is to be submitted for processing. In this embodiment of the invention, load distributor 112 is implemented in hardware, thus allowing rapid execution of the hashing and modulus functions. In an alternative embodiment of the invention, virtually any number of processors may be accommodated.

Detailed Description Text (286):
In state 718, a thread or other process on the selected processor begins processing the packet that was stored in the processor's queue. Methods of processing a packet through its protocol stack are well known to those skilled in the art and need not be described in detail. The illustrated procedure then ends with end state 720.

Detailed Description Text (371):
The illustrated embodiments of the invention employ four categories of host memory buffers, the sizes of which are programmable. The buffer sizes are programmable in order to accommodate various host platforms, but are programmed to be one memory page in size in present embodiments in order to enhance the efficiency of handling and processing network traffic. For example, the embodiments discussed in this section are directed to the use of a host computer system employing a SPARC.TM. processor, and so each buffer is eight kilobytes in size. These embodiments are easily adjusted, however, for host computer systems employing memory pages having other dimensions.

Detailed Description Text (438):
Then, in state 1654 DMA engine 120 determines whether splitting of jumbo buffers is enabled. If enabled, the header of a jumbo packet is stored in a header buffer while the packet's data is stored in one or more jumbo buffers. If not enabled, the entire packet will be stored in one or more jumbo buffers. Illustratively, splitting of jumbo packets is enabled or disabled according to the configuration of a programmable indicator (e.g., flag, bit, register) that may be set by software operating on the host computer (e.g., a device driver). If splitting is enabled, the illustrated procedure continues at state 1670. Otherwise, the procedure continues with state 1656.

Detailed Description Text (563):
Then, in state 1944, DMA engine 120 determines whether splitting of jumbo buffers is enabled. If enabled, the header of a jumbo packet is stored in a header buffer while the packet's data is stored in one or more jumbo buffers. If not enabled, the entire packet will be stored in one or more jumbo buffers. Illustratively, splitting of jumbo packets is enabled or disabled according to the configuration of a programmable indicator (e.g., flag, bit, register) that is set by software

operating on the host computer (e.g., a device driver). If splitting is enabled, the illustrated procedure continues at state 1960. Otherwise, the procedure proceeds to state 1946.

Detailed Description Text (675):
For one or more regions of packet queue 2400, an associated programmable probability indicator indicates the probability that a packet will be dropped when traffic indicator 2408 indicates that the level of traffic in the packet queue has reached the associated region. Therefore, in the illustrated embodiment probability indicator 2412 indicates the probability that a packet will be dropped while the packet queue is less than half full (e.g., when traffic indicator 2408 is located in region zero). Similarly, probability indicators 2414 and 2416 specify the probability that a new packet will be dropped when traffic indicator 2408 identifies regions one and two, respectively.

Detailed Description Text (680):
Just as a packet queue may be divided into any number of regions for purposes of the present invention, probability indicators may comprise bit masks of any size or magnitude, and need not be of equal size or magnitude. Further, probability indicators are programmable in a present embodiment, thus allowing them to be altered even during the operation of a network interface.

Detailed Description Text (682):
It will be understood that probability indicators and a counter simply constitute one system for enabling the random discard of packets in a network interface. Other mechanisms are also suitable. In one alternative embodiment, a random number generator may be employed in place of a counter and/or probability indicators to enable a random discard policy. For example, when a random number is generated, such as M, the Mth packet (or every Mth packet) after the number is generated may be dropped. Or, the random number may specify a probability of dropping a packet. The random number may thus be limited to (e.g., hashed into) a certain range of values or probabilities. As another alternative, a random number generator may be used in tandem with multiple regions or thresholds within a packet queue. In this alternative embodiment a programmable value, represented here as N, may be associated with a region or queue threshold. Then, when a traffic indicator reaches that threshold or region, the Nth packet (or every Nth packet) may be dropped until another threshold or boundary is reached.

Detailed Description Text (694):
In a present embodiment of the invention, probability indicators and/or the specifications (e.g., boundaries) into which a packet queue is partitioned are programmable and may be adjusted by software operating on a host computer (e.g., a device driver). Criteria for immunizing packets may also be programmable. Methods of discarding packets in a network interface or other communication device may thus be altered in accordance with the embodiments described in this section, even during continued operation of such a device. Various other embodiments and criteria for randomly discarding packets and/or applying criteria for the intelligent discard of packets will be apparent to those skilled in the art.

Current US Cross Reference Classification (2):
709/225

Current US Cross Reference Classification (3):
709/228

US006483804B1

(12) **United States Patent**
Muller et al.

(10) **Patent No.: US 6,483,804 B1**
(45) **Date of Patent: Nov. 19, 2002**

(54) METHOD AND APPARATUS FOR DYNAMIC PACKET BATCHING WITH A HIGH PERFORMANCE NETWORK INTERFACE

(75) Inventors: **Shimon Muller**, Sunnyvale, CA (US); **Denton E. Gentry, Jr.**, Fremont, CA (US)

(73) Assignee: **Sun Microsystems, Inc.**, Santa Clara, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/260,324**

(22) Filed: **Mar. 1, 1999**

(51) **Int. Cl.^{7}** **H04J 1/16**

(52) **U.S. Cl.** **370/230**; 370/235; 709/225; 709/228

(58) **Field of Search** 370/230, 231, 370/235, 392, 389, 225, 226, 241, 401, 428, 427, 473, 474, 394, 252, 466, 409; 709/225, 226, 235, 241, 228

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,414,704 A	5/1995	Spinney	370/60
5,583,940 A	12/1996	Vidrascu et al.	380/49
5,684,954 A	11/1997	Kaiserswerth et al.	395/200.2
5,748,905 A	5/1998	Hauser et al.	395/200.79
5,758,089 A	5/1998	Gentry et al.	395/200.64
5,778,180 A	7/1998	Gentry et al.	395/200.42
5,778,414 A	7/1998	Winter et al.	711/5
5,787,255 A	7/1998	Parlan et al.	395/200.63
5,793,954 A	8/1998	Baker et al.	395/200.8
5,870,394 A	2/1999	Oprea	370/392
5,920,705 A	* 7/1999	Lyon et al.	370/409
6,157,955 A	* 12/2000	Narad et al.	709/228

FOREIGN PATENT DOCUMENTS

EP	0 447 725	9/1991	G06F/15/16
EP	0 573 739	12/1993	H04L/12/56
EP	0 853 411	7/1998	H04L/29/06
EP	0 865 180	9/1998	H04L/12/56
WO	WO 95/14269	5/1995	G06F/7/08
WO	WO 97/28505	8/1997	G06F/13/14
WO	WO 99/00737	1/1999	G06F/13/00
WO	WO99/00945	1/1999	H04L/12/46
WO	WO99/00948	1/1999	H04L/12/56
WO	WO 99/00949	1/1999	H04L/12/56

OTHER PUBLICATIONS

Toong Shoon Chan, et al., "Parallel Architecture Support for High-Speed Protocol Processing," Feb. 1, 1997, *Microprocessors And Microsystems*, vol. 20, No. 6, pp. 325–339.

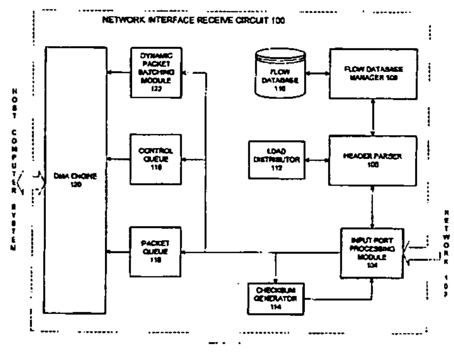
(List continued on next page.)

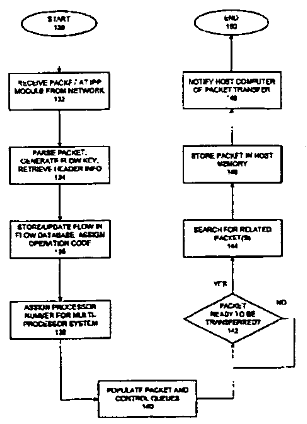
Primary Examiner—Wellington Chin
Assistant Examiner—William Schultz
(74) *Attorney, Agent, or Firm*—Park, Vaughan & Fleming LLP

(57) ABSTRACT

A system and method are provided for identifying related packets in a communication flow for the purpose of collectively processing them through a protocol stack comprising one or more protocols under which the packets were transmitted. A packet received at a network interface is parsed to retrieve information from one or more protocol headers. A flow key is generated to identify a communication flow that includes the packet, and is stored in a database of flow keys. When the packet is placed in a queue to be transferred to a host computer, the flow key and/or its flow number (e.g., its index into the database) is stored in a separate queue. Near to the time at which the packet is transferred to the host computer, a dynamic packet batching module searches for a packet that is related to the packet being transferred (i.e., is in the same flow) but which will be transferred later in time. If a related packet is located, the host computer is alerted and, as a result, delays processing the transferred packet until the related packet is also received. By collectively processing the related packets, processor time is more efficiently utilized.

27 Claims, 49 Drawing Sheets




START 130
RECEIVE PACKET AT IPP MODULE FROM NETWORK 132
PARSE PACKET; GENERATE FLOW KEY, RETRIEVE HEADER INFO 134
STORE/UPDATE FLOW IN FLOW DATABASE, ASSIGN OPERATION CODE 136
ASSIGN PROCESSOR NUMBER FOR MULTI-PROCESSOR SYSTEM 138
POPULATE PACKET AND CONTROL QUEUES 140
PACKET READY TO BE TRANSFERRED? 142
YES
NO
SEARCH FOR RELATED PACKET(S) 144
STORE PACKET IN HOST MEMORY 146
NOTIFY HOST COMPUTER OF PACKET TRANSFER 148
END 150

Freeform Search

Database:
- US Pre-Grant Publication Full-Text Database
- [✓ US Patents Full-Text Database]
- US OCR Full-Text Database
- EPO Abstracts Database
- JPO Abstracts Database
- Derwent World Patents Index
- IBM Technical Disclosure Bulletins

Term: `L5 and (select$ with thread$)`

Display: 10 **Documents in Display Format:** KWIC **Starting with Number** 1

Generate: Hit List [✓ Hit Count] Side by Side Image

Search Clear Interrupt

Search History

DATE: Sunday, May 02, 2004 Printable Copy Create Case

Set Name side by side	**Query**	**Hit Count**	**Set Name** result set
	DB=USPT; PLUR=YES; OP=ADJ		
L9	L5 and (select$ with thread$)	5	L9
L8	L6 and (select$ with thread$)	4	L8
L7	L6 and threads	8	L7
L6	L4 and (processor with programmable)	35	L6
L5	L4 and programmable	131	L5
L4	L1 and (media adj1 access adj1 controller)	244	L4
L3	L2 and MAC	8	L3
L2	L1 and (processor with (multiple or plurality) with programmable with engines)	10	L2
L1	709/$.ccls. or 712/$.ccls.	23278	L1

END OF SEARCH HISTORY